In the Claims:

Please amend Claim 13.

1. (Previously Presented): A machine-readable medium carrying one or more sequences of instructions for dynamically generating a wrapper object, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving a vendor defined class and a superclass;

performing reflection on the vendor class to obtain vendor specific extension methods defined within the vendor class the reflection including retrieving meta information for allowing a server to generate a wrapper class that matches the vendor class;

generating the wrapper class as a subclass of the superclass, wherein the wrapper class comprises at least one of the vendor specific extension methods from the vendor class;

generating a wrapper object as an instance of the wrapper class by instantiating the wrapper class, thereby associating a relationship between the wrapper object and a vendor object; and

providing the wrapper object to an application program, thereby providing the application program with access to vendor specific extension methods.

2. (Previously Presented): The machine-readable medium of claim 1 wherein the wrapper object is dynamically generated at runtime.

3. (Previously Presented): The machine-readable medium of claim 1 wherein the superclass is one of a pre-existing JDBC, JMS, or connector class.

4. (Previously Presented): The machine-readable medium of claim 1 wherein the superclass includes logic to handle server side tasks.

5. (Previously Presented): The machine-readable medium of claim 1 wherein the wrapper class is generated in bytecode.

6. (Previously Presented): The machine-readable medium of claim 5 wherein bytecode is generated for vendor methods not implemented in the superclass.

7. (Previously Presented): The machine-readable medium of claim 5 wherein the bytecode is generated using hot code generation.

8. (Previously Presented): The machine-readable medium of claim 1 wherein providing the wrapper object to an application program, enables the application program to access standard features defined by the superclass and non-standard vendor extensions defined by the vendor defined class.

9. (Previously Presented): The machine-readable medium of claim 8, wherein the standard features are J2EE features.

10. (Previously Presented): A machine-readable medium carrying one or more sequences of instructions for processing an invocation at a dynamically generated wrapper, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving, from an application program, an invocation call directed to a wrapped vendor object;

initiating pre-processing by calling a pre-invocation handler configured to execute server-side code;

calling the wrapped vendor object;

receiving a result from the wrapped vendor object;

initiating post-processing by calling a post-invocation handler configured to execute post processing server-side tasks; and

providing the result to the application program, thereby enabling the application program to access vendor specific extension methods of the wrapped vendor object.

11. – 12. (Canceled)

13. (Currently Amended): The machine-readable medium of claim ~~12~~ 10 wherein the server-side code executed by the pre-invocation handler includes global transaction processing code.

14. – 15. (Canceled)

16. (Previously Presented): The machine-readable medium of claim 10 wherein the post-processing server-side tasks include global transaction management.

17. (Previously Presented): The machine-readable medium of claim 1 wherein providing the wrapper object to an application program enables the application to access wrapped vendor objects without requiring a relinking of the application and a vendor software package.

18. (Previously Presented): The machine-readable medium of claim 10 wherein calling the wrapped vendor object enables the wrapped vendor object to be processed by the application without requiring a relinking of the application and a vendor software package.

19. – 20. (Cancelled).

21. (Previously Presented): A machine-readable medium carrying one or more sequences of instructions for enabling an application program to interface with a vendor application, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving a vendor provided class used to interface with third party software;

preparing a wrapper object for interfacing with vendor specific extension methods of the vendor provided class by reflecting the vendor provided class and a superclass to form a wrapper class from which the wrapper object is instantiated; and

providing the wrapper object to the application program, thereby enabling the application program capability to access vendor specific extension methods of the vendor application using the wrapper object.

22. (Previously Presented): A machine-readable medium carrying one or more sequences of instructions for processing an invocation at a dynamically generated wrapper enabling an application program to interface with a vendor application, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving, from an application program, an invocation call directed to a wrapped vendor object;

calling the wrapped vendor object;

receiving a result from the wrapped vendor object; and

providing the result to the application program, thereby enabling the application program to access vendor specific extension methods of the wrapped vendor object.

23. (Previously Presented): The machine-readable medium of claim 1 wherein the generating the wrapper class further includes dynamically generating the wrapper class in byte code that perfectly matches with the vendor class.

24. (Previously Presented): A machine-readable medium carrying one or more sequences of instructions for processing an invocation at a dynamically generated wrapper, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving, from an application program, a method invocation call directed to a vendor object;

calling a wrapper object for processing the method invocation call wherein the wrapper object has been dynamically generated from a vendor class to be associated with the vendor object;

initiating pre-processing by the wrapper object, wherein the wrapper object calls a pre-invocation handler configured to perform server side logic;

forwarding the method invocation call to the vendor object by the wrapper object on behalf of the application program;

receiving a result of the method invocation call from the vendor object by the wrapper object;

initiating post-processing by the wrapper object, wherein the wrapper object calls a post-invocation handler configured to perform server-side logic; and

providing the result to the application program, thereby enabling the application program to access vendor specific extension methods of the vendor object.

25. (Previously Presented): The machine-readable medium of claim 24 wherein the server-side logic includes at least one of global transaction management, pooling, caching, tracing and profiling.